



MCS[®]-51 MACRO ASSEMBLY LANGUAGE POCKET REFERENCE

TABLE OF CONTENTS

	PAGE
Program Status Word	1
Operand Definitions	1
Arithmetic Instructions	2
Logical Instructions	3
Data Move Instructions	4
Bit Manipulation Instructions	5
No Operation	5
Program and Machine Control Instructions	6
Codes for Instructions Using Registers	7
Predefined Byte Addresses	8
Predefined Code Addresses	8
Predefined Bit Addresses	9
Macro Processor Language (MPL) Functions	10
Assembly Time Expression Operators	11
Assembler Directives	11
Table of Instruction Opcodes in Hexadecimal Order	12
Hex-ASCII Table	18

PROGRAM STATUS WORD (PSW—0D0H)

C	AC	F0	RS1	RS0	OV		P
7	6	5	4	3	2	1	0

C	Carry Flag
AC	Auxiliary Carry Flag
F0	Flag 0
RS1	Register Bank Select
RS0	Register Bank Select
OV	Overflow Flag
PSW.1	Not Used
P	Parity

OPERAND DEFINITIONS

Operand	Meaning
<i># data</i>	data coded in instruction
<i>data addr</i>	on-chip memory address
<i>Rr</i>	$0 \leq r \leq 7$ General-Purpose register
<i>@Rr</i>	$0 \leq r \leq 1$ Indirect address register
<i>code addr</i>	16-bit address encoded as: full 16-bit 11-bit page address 8-bit relative offset
<i>A</i>	Accumulator
<i>C</i>	Carry flag
<i>bit addr</i>	bit address (on-chip)
<i>/ bit addr</i>	complemented contents of bit address
<i>DPTR</i>	Data Pointer
<i>PC</i>	Program Counter
<i>AB</i>	Register Pair

ARITHMETIC INSTRUCTIONS

Hex Code	Mnemonic	Operands
24	ADD	A, #data
26	ADD	A, @R0
27		R1
2*	ADD	A, Rr
25	ADD	A, data addr
34	ADDC	A, #data
36	ADDC	A, @R0
37		R1
3*	ADDC	A, Rr
35	ADDC	A, data addr
94	SUBB	A, #data
96	SUBB	A, @R0
97		R1
9*	SUBB	A, Rr
95	SUBB	A, data addr
04	INC	A
06	INC	@R0
07		R1
0*	INC	Rr
05	INC	data addr
A3	INC	DPTR
14	DEC	A
16	DEC	@R0
17		R1
1*	DEC	Rr
15	DEC	data addr
A4	MUL	AB
84	DIV	AB
D4	DA	A

*See page 7 for instruction's hex code.

LOGICAL INSTRUCTIONS

Hex Code	Mnemonic	Operands
54	ANL	A, #data
56	ANL	A, @R0
57		R1
5*	ANL	A, Rr
55	ANL	A, data addr
53	ANL	data addr, #data
52	ANL	data addr, A
44	ORL	A, #data
46	ORL	A, @R0
47		R1
4*	ORL	A, Rr
45	ORL	A, data addr
43	ORL	data addr, #data
42	ORL	data addr, A
64	XRL	A, #data
66	XRL	A, @R0
67		R1
6*	XRL	A, Rr
65	XRL	A, data addr
63	XRL	data addr, #data
62	XRL	data addr, A
E4	CLR	A
F4	CPL	A
23	RL	A
33	RLC	A
03	RR	A
13	RRC	A
C4	SWAP	A

* See page 7 for instruction's hex code.

DATA MOVE INSTRUCTIONS

Hex Code	Mnemonic	Operands
74	MOV	A,#data
E6	MOV	A,@R0
E7		R1
E*	MOV	A,Rr
E5	MOV	A,data addr
F6	MOV	@R0,A
F7		R1
76	MOV	@R0,#data
77		R1
A6	MOV	@R0,data addr
A7		R1
F*	MOV	Rr,A
7*	MOV	Rr,#data
A*	MOV	Rr,data addr
F5	MOV	data addr,A
75	MOV	data addr,#data
86	MOV	data addr,@R0
87		R1
8*	MOV	data addr,Rr
85	MOV	data addr,data addr
90	MOV	DPTR,#data ☆
93	MOVC	A,@A+DPTR
83	MOVC	A,@A+PC
E0	MOVB	A,@DPTR
E2	MOVB	A,@R0
E3		R1
F0	MOVB	@DPTR,A
F2	MOVB	@R0,A
F3		R1
C0	PUSH	data addr
D0	POP	data addr
C*	XCH	A,Rr
C5	XCH	A,data addr
C6	XCH	A,R0
C7		R1
D6	XCHD	A,@R0
D7		R1

☆ 16-bit data operand.

* See page 7 for instruction's hex code.

BIT MANIPULATION INSTRUCTIONS

Hex Code	Mnemonic	Operands
C3	CLR	C
C2	CLR	<i>bit addr</i>
D3	SETB	C
D2	SETB	<i>bit addr</i>
B3	CPL	C
B2	CPL	<i>bit addr</i>
82	ANL	C, <i>bit addr</i>
B0	ANL	C, <i>lbit addr</i>
72	ORL	C, <i>bit addr</i>
A0	ORL	C, <i>lbit addr</i>
A2	MOV	C, <i>bit addr</i>
92	MOV	<i>bit addr</i> ,C

NO OPERATION

Hex Code	Mnemonic
00	NOP

PROGRAM AND MACHINE CONTROL INSTRUCTIONS

Hex Code	Mnemonic	Operands
†1	AJMP	<i>code addr</i>
B6	CJNE	<i>@R0,#data,code addr</i>
B7		<i>R1</i>
B4	CJNE	<i>A,#data,code addr</i>
B5	CJNE	<i>A,data addr,code addr</i>
B*	CJNE	<i>Rr,#data,code addr</i>
D*	DJNZ	<i>Rr,code addr</i>
D5	DJNZ	<i>data addr,code addr</i>
20	JB	<i>bit addr,code addr</i>
10	JBC	<i>bit addr,code addr</i>
40	JC	<i>code addr</i>
73	JMP	<i>@A + DPTR</i>
30	JNB	<i>bit addr,code addr</i>
50	JNC	<i>code addr</i>
70	JNZ	<i>code addr</i>
60	JZ	<i>code addr</i>
02	LJMP	<i>code addr</i>
80	SJMP	<i>code addr</i>
†1	ACALL	<i>code addr</i>
12	LCALL	<i>code addr</i>
22	RET	
32	RETI	
**	JMP	<i>code addr</i>
**	CALL	<i>code addr</i>

† First three bits of the opcode are formed by the code address operand.

* See page 7 for instruction's hex code.

** Generic form of instruction.

CODES FOR INSTRUCTIONS USING REGISTERS

	MOV Rr,A	MOV Rr, # data	MOV A,Rr	MOV data addr,Rr
R0	F8H	78H	E8H	88H
R1	F9H	79H	E9H	89H
R2	FAH	7AH	EAH	8AH
R3	FBH	7BH	EBH	8BH
R4	FCH	7CH	ECH	8CH
R5	FDH	7DH	EDH	8DH
R6	FEH	7EH	EEH	8EH
R7	FFH	7FH	EFH	8FH
	XCH A,Rr	XRL A,Rr	INC Rr	MOV Rr, data addr
R0	C8H	68H	08H	A8H
R1	C9H	69H	09H	A9H
R2	CAH	6AH	0AH	AAH
R3	CBH	6BH	0BH	ABH
R4	CCH	6CH	0CH	ACH
R5	CDH	6DH	0DH	ADH
R6	CEH	6EH	0EH	AEH
R7	CFH	6FH	0FH	AFH
	ADD A,Rr	ADDC A,Rr	ANL A,Rr	CJNE Rr, #data, code addr
R0	28H	38H	58H	B8H
R1	29H	39H	59H	B9H
R2	2AH	3AH	5AH	BAH
R3	2BH	3BH	5BH	BBH
R4	2CH	3CH	5CH	BCH
R5	2DH	3DH	5DH	BDH
R6	2EH	3EH	5EH	BEH
R7	2FH	3FH	5FH	BFH
	ORL A,Rr	SUBB A,Rr	DEC Rr	DJNZ Rr, code addr
R0	48H	98H	18H	D8H
R1	49H	99H	19H	D9H
R2	4AH	9AH	1AH	DAH
R3	4BH	9BH	1BH	DBH
R4	4CH	9CH	1CH	DCH
R5	4DH	9DH	1DH	DDH
R6	4EH	9EH	1EH	DEH
R7	4FH	9FH	1FH	DFH

PREDEFINED BYTE ADDRESSES

Symbol	Hex Address	Meaning
ACC	E0	ACCUMULATOR
B	F0	MULTIPLICATION REGISTER
DPH	83	DATA POINTER <high byte>
DPL	82	DATA POINTER <low byte>
IE	A8	INTERRUPT ENABLE
IP	B8	INTERRUPT PRIORITY
P0	80	PORT 0
P1	90	PORT 1
P2	A0	PORT 2
P3	B0	PORT 3
PSW	D0	PROGRAM STATUS WORD
SBUF	99	SERIAL PORT BUFFER
SCON	98	SERIAL PORT CONTROL
SP	81	STACK POINTER
TCON	88	TIMER CONTROL
TH0	8C	TIMER 0 <high byte>
TH1	8D	TIMER 1 <high byte>
TL0	8A	TIMER 0 <low byte>
TL1	8B	TIMER 1 <low byte>
TMOD	89	TIMER MODE

PREDEFINED CODE ADDRESSES

Symbol	Hex Address	Meaning
RESET	00	Power Up (Reset)
EXTI0	03	External Interrupt 0
TIMER0	0B	Timer 0 Interrupt
EXTI1	13	External Interrupt 1
TIMER1	1B	Timer 1 Interrupt
SINT	23	Serial Port Interrupt

PREDEFINED BIT ADDRESSES

Sym.	Position	Hex	Meaning
CY	PSW.7	D7	CARRY FLAG
AC	PSW.6	D6	AUXILIARY CARRY FLAG
F0	PSW.5	D5	FLAG 0
RS1	PSW.4	D4	REGISTER BANK SELECT BIT 1
RS0	PSW.3	D3	REGISTER BANK SELECT BIT 0
OV	PSW.2	D2	OVERFLOW FLAG
P	PSW.0	D0	PARITY FLAG
TF1	TCON.7	8F	TIMER 1 OVERFLOW FLAG
TR1	TCON.6	8E	TIMER 1 RUN CONTROL BIT
TF0	TCON.5	8D	TIMER 0 OVERFLOW FLAG
TR0	TCON.4	8C	TIMER 0 RUN CONTROL BIT
IE1	TCON.3	8B	INTERRUPT 1 EDGE FLAG
IT1	TCON.2	8A	INTERRUPT 1 TYPE CONTROL BIT
IE0	TCON.1	89	INTERRUPT 0 EDGE FLAG
IT0	TCON.0	88	INTERRUPT 0 TYPE CONTROL BIT
SM0	SCON.7	9F	SERIAL MODE CONTROL BIT 0
SM1	SCON.6	9E	SERIAL MODE CONTROL BIT 1
SM2	SCON.5	9D	SERIAL MODE CONTROL BIT 2
REN	SCON.4	9C	RECEIVE ENABLE
TB8	SCON.3	9B	TRANSMIT BIT 8
RB8	SCON.2	9A	RECEIVE BIT 8
TI	SCON.1	99	TRANSMIT INTERRUPT FLAG
RI	SCON.0	98	RECEIVE INTERRUPT FLAG
EA	IE.7	AF	ENABLE ALL INTERRUPTS
ES	IE.4	AC	ENABLE SERIAL PORT INTERRUPT
ET1	IE.3	AB	ENABLE TIMER 1 INTERRUPT
EX1	IE.2	AA	ENABLE EXTERNAL INTERRUPT 1
ET0	IE.1	A9	ENABLE TIMER 0 INTERRUPT
EX0	IE.0	A8	ENABLE EXTERNAL INTERRUPT 0
PS	IP.4	BC	PRIORITY OF SERIAL PORT INTERRUPT
PT1	IP.3	BB	PRIORITY OF TIMER 1 INTERRUPT
PX1	IP.2	BA	PRIORITY OF EXTERNAL INTERRUPT 1
PT0	IP.1	B9	PRIORITY OF TIMER 0
PX0	IP.0	B8	PRIORITY OF EXTERNAL INTERRUPT 0
RD	P3.7	B7	READ DATA FOR EXTERNAL MEMORY
WR	P3.6	B6	WRITE DATA FOR EXTERNAL MEMORY
T1	P3.5	B5	TIMER/COUNTER 1 EXTERNAL FLAG
T0	P3.4	B4	TIMER/COUNTER 0 EXTERNAL FLAG
INT1	P3.3	B3	INTERRUPT 1 INPUT PIN
INT0	P3.2	B2	INTERRUPT 0 INPUT PIN
TXD	P3.1	B1	SERIAL PORT TRANSMIT PIN
RXD	P3.0	B0	SERIAL PORT RECEIVE PIN

MACRO PROCESSOR LANGUAGE (MPL) FUNCTIONS

DEFINING A FUNCTION

%*DEFINE (macro-name) (replacement-pattern)
%*DEFINE (macro-name(parameter-list)) (replacement-pattern)

MANIPULATING STRINGS

%EVAL(expression)
%LEN(string)
%SUBSTR(string,expr1,expr2)
%IN
%OUT(string)

CONTROL FUNCTIONS

%IF (expr) THEN (replacement-value) [ELSE
(replacement-value)]FI
%REPEAT (expr) (replacement-value)
%WHILE (expr) (replacement-value)
%SET(name, value)
%MATCH(name1, name2) (list)
%MATCH(name1 delimiter name2) (string)

INTERPRETATION-CONTROLLING FUNCTIONS

%(balanced-text)
%nxxx...x
 n
%EXIT
%'comment-text'
or:
%'comment-text linefeed
%METACHAR(char)

COMPARING STRINGS LEXICALLY

%EQS
%NES
%LTS
%LES
%GTS
%GES } (string1, string2)

ASSEMBLY TIME EXPRESSION OPERATORS

In order of decreasing precedence

()

HIGH, LOW

*, /, MOD, SHR, SHL

+, -

EQ, LT, GT, LE, GE, NE, =, <, >, <=, >=, <>

NOT

AND

OR, XOR

ASSEMBLER DIRECTIVES

Directive	Meaning
SEGMENT	Define a relocatable segment
EQU	Assign a numeric value or special assembler symbol
SET	Set symbol value
DATA	Define a data address symbol
IDATA	Define an indirect internal data address symbol
XDATA	Define an off chip data address symbol
BIT	Assign a bit address symbol
CODE	Assign a code address symbol
DS	Reserve space in byte units
DB	Insert a list of byte values
DW	Insert a list of word values
DBIT	Advance bit location counter
PUBLIC	Declare symbols outside current module
EXTRN	List symbols defined in other modules
NAME	Identify current program module
ORG	Set location counter value
END	End of program
RSEG	Select a relocatable segment
CSEG	Select an absolute segment within code address space
DSEG	Select an absolute segment within internal data address space
XSEG	Select an absolute segment within external data address space
ISEG	Select an absolute segment within indirect internal data address space
BSEG	Select an absolute segment within the bit data address space
USING	Select predefined symbolic register banks

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER

Hex Code	Number of Bytes	Mnemonic	Operands
00	1	NOP	
01	2	AJMP	<i>code addr</i>
02	3	LJMP	<i>code addr</i>
03	1	RR	A
04	1	INC	A
05	2	INC	<i>data addr</i>
06	1	INC	@R0
07	1	INC	@R1
08	1	INC	R0
09	1	INC	R1
0A	1	INC	R2
0B	1	INC	R3
0C	1	INC	R4
0D	1	INC	R5
0E	1	INC	R6
0F	1	INC	R7
10	3	JBC	<i>bit addr, code addr</i>
11	2	ACALL	<i>code addr</i>
12	3	LCALL	<i>code addr</i>
13	1	RRC	A
14	1	DEC	A
15	2	DEC	<i>data addr</i>
16	1	DEC	@R0
17	1	DEC	@R1
18	1	DEC	R0
19	1	DEC	R1
1A	1	DEC	R2
1B	1	DEC	R3
1C	1	DEC	R4
1D	1	DEC	R5
1E	1	DEC	R6
1F	1	DEC	R7
20	3	JB	<i>bit addr, code addr</i>
21	2	AJMP	<i>code addr</i>
22	1	RET	
23	1	RL	A
24	2	ADD	A, # <i>data</i>
25	2	ADD	A, <i>data addr</i>
26	1	ADD	A, @R0
27	1	ADD	A, @R1
28	1	ADD	A, R0
29	1	ADD	A, R1
2A	1	ADD	A, R2

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Cont'd.)

Hex Code	Number of Bytes	Mnemonic	Operands
2B	1	ADD	A,R3
2C	1	ADD	A,R4
2D	1	ADD	A,R5
2E	1	ADD	A,R6
2F	1	ADD	A,R7
30	3	JNB	<i>bit addr,code addr</i>
31	2	ACALL	<i>code addr</i>
32	1	RETI	
33	1	RLC	A
34	2	ADDC	A,#data
35	2	ADDC	A,data addr
36	1	ADDC	A,@R0
37	1	ADDC	A,@R1
38	1	ADDC	A,R0
39	1	ADDC	A,R1
3A	1	ADDC	A,R2
3B	1	ADDC	A,R3
3C	1	ADDC	A,R4
3D	1	ADDC	A,R5
3E	1	ADDC	A,R6
3F	1	ADDC	A,R7
40	2	JC	<i>code addr</i>
41	2	AJMP	<i>code addr</i>
42	2	ORL	<i>data addr,A</i>
43	3	ORL	<i>data addr,#data</i>
44	2	ORL	A,#data
45	2	ORL	A,data addr
46	1	ORL	A,@R0
47	1	ORL	A,@R1
48	1	ORL	A,R0
49	1	ORL	A,R1
4A	1	ORL	A,R2
4B	1	ORL	A,R3
4C	1	ORL	A,R4
4D	1	ORL	A,R5
4E	1	ORL	A,R6
4F	1	ORL	A,R7
50	2	JNC	<i>code addr</i>
51	2	ACALL	<i>code addr</i>
52	2	ANL	<i>data addr,A</i>
53	3	ANL	<i>data addr,#data</i>
54	2	ANL	A,#data
55	2	ANL	A,data addr

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Cont'd.)

Hex Code	Number of Bytes	Mnemonic	Operands
56	1	ANL	A,@R0
57	1	ANL	A,@R1
58	1	ANL	A,R0
59	1	ANL	A,R1
5A	1	ANL	A,R2
5B	1	ANL	A,R3
5C	1	ANL	A,R4
5D	1	ANL	A,R5
5E	1	ANL	A,R6
5F	1	ANL	A,R7
60	2	JZ	<i>code addr</i>
61	2	AJMP	<i>code addr</i>
62	2	XRL	<i>data addr</i> ,A
63	3	XRL	<i>data addr</i> ,# <i>data</i>
64	2	XRL	A,# <i>data</i>
65	2	XRL	A, <i>data addr</i>
66	1	XRL	A,@R0
67	1	XRL	A,@R1
68	1	XRL	A,R0
69	1	XRL	A,R1
6A	1	XRL	A,R2
6B	1	XRL	A,R3
6C	1	XRL	A,R4
6D	1	XRL	A,R5
6E	1	XRL	A,R6
6F	1	XRL	A,R7
70	2	JNZ	<i>code addr</i>
71	2	ACALL	<i>ccode addr</i>
72	2	ORL	C, <i>bit addr</i>
73	1	JMP	@A + DPTR
74	2	MOV	A,# <i>data</i>
75	3	MOV	<i>data addr</i> ,# <i>data</i>
76	2	MOV	@R0,# <i>data</i>
77	2	MOV	@R1,# <i>data</i>
78	2	MOV	R0,# <i>data</i>
79	2	MOV	R1,# <i>data</i>
7A	2	MOV	R2,# <i>data</i>
7B	2	MOV	R3,# <i>data</i>
7C	2	MOV	R4,# <i>data</i>
7D	2	MOV	R5,# <i>data</i>
7E	2	MOV	R6,# <i>data</i>
7F	2	MOV	R7,# <i>data</i>
80	2	SJMP	<i>code addr</i>

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Cont'd.)

Hex Code	Number of Bytes	Mnemonic	Operands
81	2	AJMP	<i>code addr</i>
82	2	ANL	<i>C, bit addr</i>
83	1	MOVC	<i>A, @A + PC</i>
84	1	DIV	<i>AB</i>
85	3	MOV	<i>data addr, data addr</i>
86	2	MOV	<i>data addr, @R0</i>
87	2	MOV	<i>data addr, @R1</i>
88	2	MOV	<i>data addr, R0</i>
89	2	MOV	<i>data addr, R1</i>
8A	2	MOV	<i>data addr, R2</i>
8B	2	MOV	<i>data addr, R3</i>
8C	2	MOV	<i>data addr, R4</i>
8D	2	MOV	<i>data addr, R5</i>
8E	2	MOV	<i>data addr, R6</i>
8F	2	MOV	<i>data addr, R7</i>
90	3	MOV	<i>DPTR, #data</i>
91	2	ACALL	<i>code addr</i>
92	2	MOV	<i>bit addr, C</i>
93	1	MOVC	<i>A, @A + DPTR</i>
94	2	SUBB	<i>A, #data</i>
95	2	SUBB	<i>A, data addr</i>
96	1	SUBB	<i>A, @R0</i>
97	1	SUBB	<i>A, @R1</i>
98	1	SUBB	<i>A, R0</i>
99	1	SUBB	<i>A, R1</i>
9A	1	SUBB	<i>A, R2</i>
9B	1	SUBB	<i>A, R3</i>
9C	1	SUBB	<i>A, R4</i>
9D	1	SUBB	<i>A, R5</i>
9E	1	SUBB	<i>A, R6</i>
9F	1	SUBB	<i>A, R7</i>
A0	2	ORL	<i>C, 1 bit addr</i>
A1	2	AJMP	<i>code addr</i>
A2	2	MOV	<i>C, bit addr</i>
A3	1	INC	<i>DPTR</i>
A4	1	MUL	<i>AB</i>
A5		reserved	
A6	2	MOV	<i>@R0, data addr</i>
A7	2	MOV	<i>@R1, data addr</i>
A8	2	MOV	<i>R0, data addr</i>
A9	2	MOV	<i>R1, data addr</i>
AA	2	MOV	<i>R2, data addr</i>
AB	2	MOV	<i>R3, data addr</i>

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Cont'd.)

Hex Code	Number of Bytes	Mnemonic	Operands
AC	2	MOV	R4,data addr
AD	2	MOV	R5,data addr
AE	2	MOV	R6,data addr
AF	2	MOV	R7,data addr
B0	2	ANL	C,1bit addr
B1	2	ACALL	code addr
B2	2	CPL	bit addr
B3	1	CPL	C
B4	3	CJNE	A,#data ,code addr
B5	3	CJNE	A,data addr ,code addr
B6	3	CJNE	@R0,#data ,code addr
B7	3	CJNE	@R1,#data ,code addr
B8	3	CJNE	R0,#data ,code addr
B9	3	CJNE	R1,#data ,code addr
BA	3	CJNE	R2,#data ,code addr
BB	3	CJNE	R3,#data ,code addr
BC	3	CJNE	R4,#data ,code addr
BD	3	CJNE	R5,#data ,code addr
BE	3	CJNE	R6,#data ,code addr
BF	3	CJNE	R7,#data ,code addr
C0	2	PUSH	data addr
C1	2	AJMP	code addr
C2	2	CLR	bit addr
C3	1	CLR	C
C4	1	SWAP	A
C5	2	XCH	A,data addr
C6	1	XCH	A,@R0
C7	1	XCH	A,@R1
C8	1	XCH	A,R0
C9	1	XCH	A,R1
CA	1	XCH	A,R2
CB	1	XCH	A,R3
CC	1	XCH	A,R4
CD	1	XCH	A,R5
CE	1	XCH	A,R6
CF	1	XCH	A,R7
D0	2	POP	data addr
D1	2	ACALL	code addr
D2	2	SETB	bit addr
D3	1	SETB	C
D4	1	DA	A
D5	3	DJNZ	data addr ,code addr
D6	1	XCHD	A,@R0

TABLE OF INSTRUCTION OPCODES IN HEXADECIMAL ORDER (Cont'd.)

Hex Code	Number of Bytes	Mnemonic	Operands
D7	1	XCHD	A,@R1
D8	2	DJNZ	R0, <i>code addr</i>
D9	2	DJNZ	R1, <i>code addr</i>
DA	2	DJNZ	R2, <i>code addr</i>
DB	2	DJNZ	R3, <i>code addr</i>
DC	2	DJNZ	R4, <i>code addr</i>
DD	2	DJNZ	R5, <i>code addr</i>
DE	2	DJNZ	R6, <i>code addr</i>
DF	2	DJNZ	R7, <i>code addr</i>
E0	1	MOVB	A,@DPTR
E1	2	AJMP	<i>code addr</i>
E2	1	MOVB	A,@R0
E3	1	MOVB	A,@R1
E4	1	CLR	A
E5	2	MOV	A, <i>data addr</i>
E6	1	MOV	A,@R0
E7	1	MOV	A,@R1
E8	1	MOV	A,R0
E9	1	MOV	A,R1
EA	1	MOV	A,R2
EB	1	MOV	A,R3
EC	1	MOV	A,R4
ED	1	MOV	A,R5
EE	1	MOV	A,R6
EF	1	MOV	A,R7
F0	1	MOVB	@DPTR,A
F1	2	ACALL	<i>code addr</i>
F2	1	MOVB	@R0,A
F3	1	MOVB	@R1,A
F4	1	CPL	A
F5	2	MOV	<i>data addr</i> ,A
F6	1	MOV	@R0,A
F7	1	MOV	@R1,A
F8	1	MOV	R0,A
F9	1	MOV	R1,A
FA	1	MOV	R2,A
FB	1	MOV	R3,A
FC	1	MOV	R4,A
FD	1	MOV	R5,A
FE	1	MOV	R6,A
FF	1	MOV	R7,A

HEX-ASCII TABLE

00	NUL	2B	+	56	V
01	SOH	2C	,	57	W
02	STX	2D	-	58	X
03	ETX	2E	.	59	Y
04	EOT	2F	/	5A	Z
05	ENQ	30	0	5B	[
06	ACK	31	1	5C	\
07	BEL	32	2	5D]
08	BS	33	3	5E	^ (↑)
09	HT	34	4	5F	_ (←)
0A	LF	35	5	60	`
0B	VT	36	6	61	a
0C	FF	37	7	62	b
0D	CR	38	8	63	c
0E	SO	39	9	64	d
0F	SI	3A	:	65	e
10	DLE	3B	;	66	f
11	DC1 (X-ON)	3C	<	67	g
12	DC2 (TAPE)	3D	=	68	h
13	DC3 (X-OFF)	3E	>	69	i
14	DC4 (TAPE)	3F	?	6A	j
15	NAK	40	@	6B	k
16	SYN	41	A	6C	l
17	ETB	42	B	6D	m
18	CAN	43	C	6E	n
19	EM	44	D	6F	o
1A	SUB	45	E	70	p
1B	ESC	46	F	71	q
1C	FS	47	G	72	r
1D	GS	48	H	73	s
1E	RS	49	I	74	t
1F	US	4A	J	75	u
20	SP	4B	K	76	v
21	!	4C	L	77	w
22	"	4D	M	78	x
23	#	4E	N	79	y
24	\$	4F	O	7A	z
25	%	50	P	7B	{
26	&	51	Q	7C	
27	'	52	R	7D	} (ALT MODE)
28	(53	S	7E	~
29)	54	T	7F	DEL (RUB OUT)
2A	*	55	U		

intel®

3065 Bowers Avenue, Santa Clara, California 95051

(408) 987-8080

Printed in U.S.A.

MICROCONTROLLER

0591/3.1K/0785/SOS/AD